Trust Management within Virtual Communities: Adaptive and Socially-Compliant Trust Model

Reda Yaich, Olivier Boissier, Philippe Jaillon, Gauthier Picard

Trust Management within Virtual Communities: Adaptive and Socially-Compliant Trust Model

R. Yaich, O. Boissier, P. Jaillon and G. Picard

Abstract

Recent years have witnessed increasing interest of people in sharing, collaborating and interacting in many different ways among new social structures called Virtual Communities (VC). They represent aggregations of entities with common interests, goals, practices or values. VCs are particularly complex environments wherein trust became, rapidly, a prerequisite for the decision-making process, and where traditional trust establishment techniques are regularly challenged. In our work we are considering how individual and collective trust policies can be managed, adapted and combined. To this aim, we propose an Adaptive and Socially-Compliant Trust Management System (ASC-TMS) based on multi-agent technologies. In this framework, policies are used as concrete implementations of trust models in order to specify both (i) user-centred (i.e. personal) and community-centred (i.e. collective) trust requirements. Agents are used to manage and combine these different policies in a decentralized and flexible way. We describe the functionalities and the architecture that supports them and discuss also a prototype implementation.

1 Introduction

1.1 Context

Online environments such as virtual communities are known to be complex structures (open, dynamic, uncertain and risky) where trust became rapidly a prerequisite for the decision-making process. Further, several characteristics of these environments are constantly challenging our societies on how trust is produced and assessed; besides identifying key trust factors, finding the appropriate formalism to represent trust requirements has been, for the last decade, one of the most interesting issues for both industrials and academics.

Trust policies [18] are considered to be a viable solution to represent trust requirements. They constitute a flexible way to specify under which conditions trust may be granted. However, none of the existing languages is able to address properly new considerations raised by VCs.

1.2 Problematic

One of the main issues in specifying and deploying trust policies in VC is the ability to manage unpredictability that is inherent to their dynamic and evolving character. Unforeseen changes must be handled otherwise running policies could rapidly become obsolete and inefficient. Another important issue is the capacity of personal policies, specified individually by entities, to integrate seamlessly collective trust requirements that could be imposed or recommended by virtual communities or virtual organizations. In fact, humans in their daily decisions often consider trust by combining, at the same time, two requirements' levels: Collective and personal requirements. Collective requirements represent common trust criteria that should/must be used by the members of the same community/organization, while personal requirements reflect personal and subjective trust criteria. Thus, it's becoming increasingly evident that neither static trust models nor inert trust policies tackles appropriately the previously cited considerations. Whether for tuning used policies to fit environment changes or for making individual requirements comply with collective ones, enabling trust policies with adaptation features is clearly needed as it has been pointed out by some authors.

1.3 Proposition

In this report, we focus on endowing Trust Management Systems with adaptation features based on rich, dynamic and integrative trust model. By rich we mean that trust is built upon a wide range of trust factors. By dynamic we mean that policies are constantly adapted and evolve over time. By combinative we mean that individual policies are combined with collective ones to yield global trust requirements. To that aim, we propose to use Policy-based Trust models[18] wherein policies are used to specify trust requirements. In order to express and illustrate mentioned aspects within our framework, we used Jason [7], an extension of AgentSpeak[15] syntax, as the underlying logic formalism. The remainder of this report is organized as follows. The next section introduces background concepts including details on trust basis. To illustrate our approach we use a simple virtual community scenario wherein requirements has been identified. Section 3 introduces the foundations of our approach, including the system overview and the trust model outline. Section 4 introduces the details on policies representation. The framework architecture is illustrated and discussed in Section 5. Finally, concluding remarks and future works are presented in Section 6.

2 Background And Positioning

The last decade has seen an increasing interest on trust that gave rise to numerous researches and many definitions. In this report, we do not consider investigating these definitions nor do we attempt to add a new one. We refer readers to [3] that summarizes the existing literature. However, given the multidimensional nature of trust, we are more interested in identifying essential trust factors (i.e. sources of information) based on which trust is built. To that aim, we will have a quick insight on existing trust models in order to isolate pertinent trust factors.

2.1 Trust Models

Among the large literature on trust, sociologists vision is particularly interesting. They identified two kinds of factors that are widely used in trust assessment among human societies: *hard trust* and *soft trust* factors[11], where hard trust factors represent information derived from security mechanisms such as identity *keys*, *credentials* and *certificates*, whereas soft trust factors encompass information that are inferred from experience and observations of others. Based on this, existing trust models can be categorized into two families: hard trust models and soft trust models.

2.2 Hard trust models

They focus on managing digitally signed documents (e.g. public keys, credentials and certificates) and enforcing policies. These approaches assume, in most of cases, that trust is derived by obtaining sufficient credentials that satisfy the policy. PGP [1], PolicyMaker [4] and IBM TE [10] are some of the works, among others, that pioneered this type of models. These models highlights some common features of trust: it is direct and is accorded based on the identity key [1, 4], transferable by authorization credentials [4], and/or subjective and depends on different properties certified by a third party authority [10].

2.3 Soft trust Models

They mainly stress the importance of the social dimension of trust [8]. These models build trust based on others history [12] or evaluations[17, 11] in absence of (or in addition to) personal ones. The history represents the experience over past interactions while reputations are measures of trust associated to individuals. Both factors are used based on direct and indirect information where personal experience and assessments are internally computed and indirect information are aggregated over all the community (i.e. social structure).
Full integration of soft and hard trust factors within trust models has not

been done yet. However, [5] and [13] paved the way recently for such perspective by exploring the combination of these two types of trust factors into what we call *Hybrid Trust Models*. They represent new approachs that make use of both hard and soft trust factors. In addition to hard an soft trust factors, these models, often, consider the use of any other factor that may be pertinent for trust evaluation and open doors for any source of information including context information the risk and the outcome of an interaction. This work is an additional right step toward the proposition of a better integration of hard and soft trust factors in more realistic and rich model. For that, we consider the use of a wide range of source of information. Without loss of generality, chosen factors have been limited to the most commonly used ones among reviewed models.

2.4 Trust issues in virtual communities

In this section, we will explore first some interesting issues that are challenging trust establishment within VC and have not been solved yet. To that aim, we support our motivation with three scenarios of Virtual Communities. We then used this scenarios in order to derive and argue a set of important features that should be addressed by modern trust management models.

1. **Developers Community** Alice is regularly developing new Android applications within the Google Android Community (GAC). Before releasing the final version, Alice often needs the help of some GAC members for pointing out and fixing bugs. However, granting access to her project repository is risky (e.g. the code could be revealed and the binaries distributed). Thus, Alice limits the access to her code (resp. binary) to qualified members (i.e. having competences in Android/Java) that have reputation higher than 0,7 (resp. 0,6). As a member of the GAC, Alice is also obliged to comply with the model used by the community and must rely on identity (e.g. PGP keys) as an additional trust criterion.

2. **Scientific Virtual Community:** Bob is a PhD student in the LSTI lab. He is also member of the ePub Community where he can exchange with the other members scientific articles. Bob is disposed to share his articles only with student members having a good reputation (higher than 0,6). The ePub community members are copyright conscious and agreed that they must respect the publishers policies while sharing articles. Thus, articles access is granted only to the members that are authorized by the publisher of the article.

3. **Open Innovation Community:** Hypios is a new open innovation platform whose users form a widespread community. Hypios proposes

a showcase for R&D problems to which the community members try to find a solution and earn money for that. Carol, as an avid passionate of challenging problems, joined recently Hypios. She first managed to solve individually some simple problems but for the actual one she requires the help of some mathematicians. She created a group in order to solve collaboratively the problem and share the reward. Carol restricts the access to her group to members with mathematic backgrounds and a reputation of at least 0,7. Moreover, some problems are sensitive and the access to their description should be protected by respecting both Hypos and companies trust requirements. For instance, in order to exclude malevolent members, Hypios indicates to it's members the least tolerated value for the reputation. Thus, members reputation threshold must be at least equal to the advocated value (e.g. 0.6).

Each of the above scenarios highlights a number of interesting characteristics that should be addressed by modern trust management models. We focus here principally on three of them that we consider as central in our approach:
1. Subjectivity: Virtual communities, as the one presented above, are highly dynamic environments where no generic interaction pattern could be identified. Each specific interaction requiring trust establishment is different and will need specific set of trust requirements. We think that an efficient trust model should not neglect any trust factor in order to enable the user to protect himself in any circumstance.

Example 1 *Alice, Bob and Dave expressed differently their trust requirements characterizing the subjectivity of trust*

1. Dynamicity: Trust perception is not static and changes depending on the situation. However, we think that trust dynamics is not solely due to perception changes but also to how these perceptions are managed. In other words, not only trust the set of used trust factors varies, the variation comes also from how they are used and aggregated. Thus, trust models should be malleable at will.

Example 2 *Carol received three requests to join her group from, respectively, Dave (Rep = 0,75), Paul (Rep = 0,60) and Walter (Rep = 0,65). She accepted Dave's requests and refused Paul's and Walters ones. After refusing Pauls and Walter's requests due insufficient reputation, Carol was worried about the success of her objectives, as she still requires the collaboration of some mathematicians. she decided then to reduce the reputation threshold to 0,65 and finished by accepting Walter too. The dynamic of her evaluation is not due to the input information she used that remained unchanged, but its related to her perception of the environment that led her to change the way she considers these information (e.g. As the community is*

new, she considers that 0,6 is a fair value while she still respect the least tolerated value advocated by the community).

2. Social-Context Awareness: Trust is closely tied to beliefs, values, traditions, and norms of the society. The social dimension of trust received considerable attention recet year but wasn't completely explored leading to closed trust models (i.e. insensitive to social influences). We think that, like all our practices, trust is highly influenced by the social context we belong to. This influence may be strict such as the law that obliges a person to let a policeman enter his home, but in most of case is seamless and constitute a kind of inspiration.

Example 3 *Both Alice, Bob and Dave integrated social trust requirements along with their individual ones for the trust assessment.*

2.4.1 Desiderata

Our study of the above use cases led us to identify three important requirements for trust management within virtual communities. these requirements should be considered as an extension of the more traditional requirements for trust management that has already been raised in the literatures [18, 20]. Thus, in this report we are more interested in :

1. **Identifying essential trust factors (i.e. sources of information) based on wich VC members could build trust.** We are particularly motivated by stressing the importance of two kinds of factors that are widely used in human societies: *hard trust* and *soft trust* factors[11], where hard trust factors represent information derived from security mechanisms such as *keys*, *credentials* and *certificates*, whereas soft trust factors encompass information that are inferred from experience and observations of others.

2. **Specifying how these factors are aggregated in trust assessment.** A lot of works tried mimic humans behaviour by capturing the subjective aspect trust evaluations. Nevertheless, in most of the cases, the subjectivity refers only to some kind of variations in the inputs/ outputs of the trust evaluation mechanism. Further, a direct implication from the previous point, which is stressed here, is the importance of models heterogeneity which may exists from a community member to another and even from a community to another. This heterogeneity implies the coexistence of different trust models within the same environment. The question that arises here is how community members deal with such heterogeneity when they have a fixed trust model. In other words, do they need to use always the same trust model regardless of the partner, the community or the situation? For instance, what could an individual with an exclusively reputation-based trust

model do when he moves from collaborative and altruist community to a dishonest and selfish one? Thus, trust models are fare to be inert. they are in fact perpetually re-shaped to fit newly considered trust criteria. None of the cited works address this dynamic aspect of trust models where the set of used trust factors; the required values and their weight within the trust assessment are constantly reconsidered.

3. **Studying the co-influence of individual trust models on collective ones and vice-versa.** A trust model reflects minimal conditions an entity requires in order to grant trust to another entity. These entities are either atomic and represent individuals (e.g agent) or aggregated and represent social structures (e.g. communities). The influence of the community on the individual as the influence of the individual on the society has already been studied for general decision-making process. However, to the best of our knowledge, no work tried to address the effect of such co-influence in regard to trust practices. Indeed, on one hand, the community to which we belong considerably affects our trust decisions as the way we are evaluating trust. On the other hand, collective trust requirements constitute an image of individual practices leading to a mutual influence between the individual and the social trust mechanisms.

3 Foundations

In this section, we describe the basic foundations and concepts on which the proposal is built.

3.1 System Overview

The system model (cf. Fig. 1) represents a multi-agent system S which is defined by $S = \langle \mathcal{C}, \mathcal{A}, \mathcal{R}, \mathcal{O}, \mathcal{I}, \mathcal{F} \rangle$. Where $\mathcal{C}$ is a set of communities c, $\mathcal{A}$ is a set of agents a, $\mathcal{R}$ is a set of resources r, $\mathcal{O}$ is a set of operations o, $\mathcal{I}$ is a set of interactions $\imath$ taking place between agents and $\mathcal{F}$ is an ontology of trust factors f among S.

Definition 1 (Communities) *Agents from S with common interests join together to form communities wherein they engage in frequent interactions. Each community is governed under a set of* social policies *that describes minimal conditions community members must use in their trust assessment. $\forall c \in \mathcal{C}$, c is represented by $\langle \varepsilon_c^{\mathcal{C}}, \Pi_c \rangle$ where: $\varepsilon_c^{\mathcal{C}}$ is unique identifier and Π_c is the set of social policies.*

Example 4 *With respect to the motivating scenario. The Android Developers platform* DevCo *constitutes the system wherein each project represents a community. For instance, c is a community by the agent Bob.*

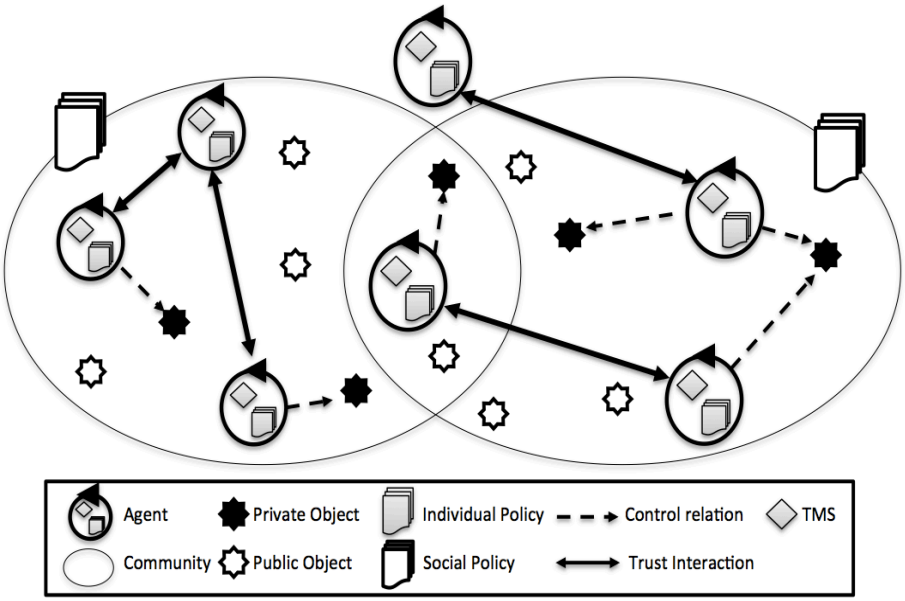

Figure 1: System model overview

Definition 2 (Agents) *Agents are autonomous entities, able to perform operations on resources. By autonomous we mean that no agent has direct control over operations of another agent. Each agent operates on the behalf of the user he is representing and integrates as a Trust Management System (TMS) protecting his private resources through trust-based decisions making process.* $\forall a \in \mathcal{A}$, $a = \langle \varepsilon_a^{\mathcal{A}}, \kappa_a, \omega_a, \beta_a, \psi_a, \mathcal{TP}_a \rangle$, *where,* $\varepsilon_a^{\mathcal{A}}$ *is the agent's unique identifier,* $\kappa_a \subseteq 2^{\mathcal{O}}$ *is the set of operations that a can perform (i.e. capabilities),* $\omega_a \subseteq 2^{\mathcal{R}}$ *is the set of resources under control of a,* $\beta_a \subseteq 2^{\mathcal{O}}$ *are communities a belongs to,* $\psi_a \subseteq 2^{\mathcal{R} \times 2^{\mathcal{O}} \times 2^{\mathcal{O}}}$ *are rights agent a acquired that state which operations a is allowed to perform on each resource, and* $\mathcal{TP}_a$ *is set of trust patterns and* $\forall tp \in \mathcal{TP}_a, tp = p_a \cup p_c \cup \in \mathcal{MP}_a$ *where* $p_a \in \pi_a$ *is and individual policy among the set* π_a *of all individual policies that has a,* $p_c \in \Pi_c$ *is the social policy of the community to which a belongs (i.e. the community c), and* $\mathcal{M}(p_a)$ *is a set of meta-policies over the policy* p_a

Definition 3 (Resources) *Resources are passive artefacts in S on which operation can be performed.* $\forall r \in \mathcal{R}$, r *is represented by* $\langle \varepsilon_r^{\mathcal{R}}, \tau_r, \varsigma_r \rangle$ *where:* $\varepsilon_r^{\mathcal{R}}$ *is its unique identifier, (ii)* τ_r *represents its type,* $\tau_{\mathcal{R}} \in \{object, service, data, credential\}$ *and* ς_r *qualifies its sensitivity,* $\varsigma_r \in \{private, public\}$. *Public resources could be manipulated by any agent without restrictions, while private resource manipulation is limited to trustworthy agents and requires permission from an agent controlling them.*

Example 5 *Bob considers online-demo as* service*, the binary of the application as an* object*, its source-code, the documentation wiki and the discussion boards as* data *while the members' electronic addresses and their PGP keys represents* credentials*. All these resources are* private *except the discussion boards that are* public*.*

Definition 4 (Operations) *Operations represent the actions agents are able to perform on a resource. Without loss of generality, we can classify operations into types.* $\forall o_i \in \mathcal{O}$, $o_i \in \tau_{\mathcal{O}} = \{access, retrieve, alter, release, grant, release, delegate\}$.

Definition 5 (Interactions) *Interactions are message exchanged between agents. Each interaction* $\imath \in \mathcal{I}$ *is defined by* $\langle a_s, a_t, \tau_\imath, \theta \rangle$*, where* a_s, a_t *are respectively the source and target agent of the message,* $\tau_\imath \in \{request, inform, reply\}$ *is the interaction type and* θ *the content of the message and* $\theta \in \mathcal{R}$*. Our focus will essentially be on requests where each request concerns a specific type of resources* τ_r *(with* $r \in \mathcal{R}$*) and a specific type of operations* τ_o *(with* $o \in \mathcal{O}$*). An interaction starts when an agent* $a_r \in \mathcal{A}$ *(called the requester) sends a request to another entity* $a_c \in \mathcal{A}$ *(called the controller) asking him permission to preform an action on a private resource* a_c *controls.*

Definition 6 (Permissions) *A permission issued from an agent* a_c *to an agent* a_r *is a declaration stating which rights* a_c *possesses with respect to a specific resource* r*. Permissions are defined by* $\langle a_c, a_r, r, \mathcal{O}^+, \mathcal{O}^- \rangle$ *where* $a_c, a_r \in \mathcal{A}$ *are, respectively, the issuer (i.e the controller) and the recipient (i.e. requester) of the permission, while* $O^+, O^- \in \mathcal{O}$ *are, respectively, sets of allowed and forbidden operations over the resource* r*.*

Now that the system model been introduced, we will describe how trust is built within such system by introducing our Trust Model.

3.2 Trust Model outline

Figure 2, illustrates our operational model for building trust and gives a temporal vision about interactions between required concepts and the process handling them. This model is used to be implemented by each agent a of the system S.

3.2.1 Model description

Within the model, agent a evaluates the trustworthiness of a requester b (where $a, b \in \mathcal{A}$) based on a trust model resulting from the combination of its individual trust requirements and the social ones. Both requirements are concretely represented by use of trust policies. **Individual policies (π)** constitute the agent's requirements in regard of according trust. These

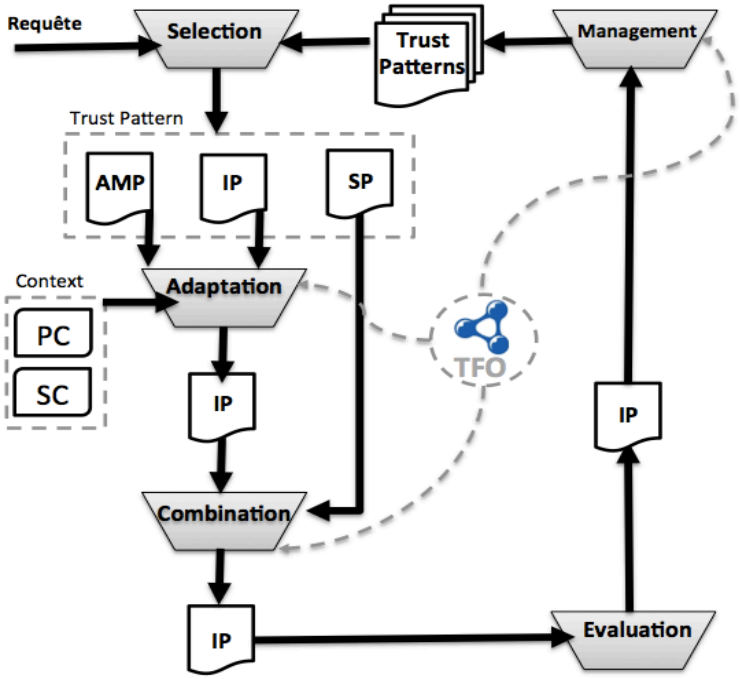

Figure 2: The Trust Model

requirements are stated by specifying restrictions on the concepts f of the Trust Factors Ontology $\mathcal{F}$ (cf. section 3.3). **Social policies** (Π) represent trust requirements imposed by the community (c here). They are used to ensure minimal trust level for each interaction undertaken within the community. **Meta-policies** ($\mathcal{M}(p)$) enable the user to specify how its individual policies p could be adapted in order to handle specified events. **Trust Patterns** ($\mathcal{TP}$) have been introduced in the model in order to reduce complexity and handle properly each type of request. In fact, as already mentioned, requests are categorized into types and each request r is mapped into a unique type τ_r. Thus, the model enables, for instance, the agent a to associate to each type $t \in \tau_r \subset (\tau_{\mathcal{O}} \times \tau_{\mathcal{R}})$ an individual policy p_a^t, a social policy p_c^t relative to the community to which a belongs and a set of meta-policies $\mathcal{M}(p_a^t)$ in order to adapt his individual policy p_a^t. Trust patterns ($\mathcal{TP}_a^t$) constitute an elegant way to structure and encapsulate policies and meta-policies based on the request type they handle. Thus, trust patterns reduce the complexity when specifying and managing trust policies while limiting the aftereffect of an error when updating a policy. As depicted above, only the appropriate trust pattern is selected and used avoiding accidents. **Context** information are captured by meta-policies and handled by triggering policies adaptation. In our approach we principally used, but are not limited to, the social and individual contexts. **Social Context (SC)** has a great

influence on trust decisions, whether the context is highly competitive or cooperative, different trust requirements can be formulated affecting the way decisions are taken. This model enables agents to make context-aware trust decision based on such indicators. Thus, information such as the number or agents are proposing the same resource (#providers) or the number of agents looking for it (#requesters) constitute good examples of such social context information. **Personnal Contex (PC)** are indicators agents uses for adapting individual policies in order to fit user (i.e. represented by the agent) requirements. Whether the requested resource is extremely sensitive or its realease is highly rewarding, such information gives the agent indicators on what to focus on: protecting the resource or releasing it. Concretely, in the model, the individual policies are regularly weakened on strengthened basede on risk and reward information for each resource and stated within the user preferences.

Example 6 (Context) *Bob is exlusively proposing a new performance testing service for Android phones. As his service is sensitive to denial-of-service attacks, he is using a policy that regulates the access to its service based on users' past behaviour. He fixed a bad-experience threshold to zero in order to exclude agents that misbehaved at least one time by making abusive use of his service. However, bob is also conscious that the community is open and at each moment new agents may propose the same service, he is also sensitive to the number of agents using his service. For that, his decided to enable policy adaptation based on his social context.*

3.2.2 Model features

The model outlined above offers three interesting features that, we believe, deserves to be highlighted: it is *rich, dynamic* and *combinative* (i.e. socially compliant). (i) *Richness* is relative to the subjectivity and the multidimensionality of used trust factors in order to build trust. (ii) *Dynamic* refers to the fact that it is constantly reshaped as it is illustrated in the figure wherein individual policies evolve over time by integrating individual preferences, social context and social policies. Finally (iii) the model is *combinative* in the sense that for each trust decision, the social requirements in regard trust evaluation are integrated to individual ones leading to versatile and socially compliant trust model. The combination of these features enables agents to benefit from a more realistic trust model where they can build, reshape and combine trust requirements at will.

3.3 Trust Factors Ontology

In the model characterized above, the trust model represents a specification of trust factors aggregation function. For that, the Trust Factors Ontology (TFO) (see Fig. 3) is central to our approach. It is used to capture and

represents essential requirements that constitute the basis for establishing trust between agents.

3.3.1 Toward semantically enabled trust models

According to [9], an ontology represent a specification of a shared conceptualization. The conceptualization refers to the choices on the way to describe a domain (e.g. trust requirements), the specification represents its formal description, while shared means that these conceptualization specification is shared among the members of a system. Ontologies are particularly interesting for modern trust management within virtual communities (and multi-agent systems as well) for several reasons. (i) First, in order to establish trust, a requester is required to provide some specific information that satisfies the controller trust requirements. These information and their semantics should be unambiguous and shared among agents of the system. Second (ii), agents within a community are asked to combine social trust requirements with their individual ones. Such combination would not be possible without a common description of these requirements. (iii) Further, agents who newly joined community may also need to discover what kinds of requirements they could use and which values they should accept for their trust assessment, the need for a semantically rich description of the trust factors arise again here. Also (iv), the structure representing trust requirements should extensible as new requirements could/should be added at need. Finally, in virtual communities where individual is constantly reshaped by social influences, ontology is crucial for agents with different trust models in order to communicate and share trust information (e.g. reputations and experiences). Therefore, the need for a formal, semantic and extensible representation of trust factors is leading the ontologies to be preferred to other representations.

3.3.2 TFO description

Different from existing ontologies [2], TFO concepts are derived from literature review and analysis of existing trust models (see section 2). Most of these models, however, emphasize some factors and abstracts away others, while it is assumed that the integration of various factors provides agent with optimal set of trust requirements [14]. Therefore and as outlined in section 2, the Trust Factors considered in the TFO as sources of information for trust evaluation are not limited. We refer instead to any factor that may be pertinent for trust evaluation. In this way, doors remain wide open for any sources of information for trust decisions to be further added.

Definition 7 (Trust Factor) *A trust factor $f_i \in \mathcal{F}$ is a tuple (τ_f, Δ_f) where τ_f is the type of the f_i and Δ_f its domain (i.e. the set of values f_i can take).*

Example 7 (Trust Factor) $(reputation, +0.6)$ *and* $(identity, "Trusted")$ *are examples of trust factors used by* DevCo *are using.*

The whole set of trust factors types ($\tau_f \in \mathcal{F}$) is hierarchically structured and forms the Trust Factors Ontology, as illustrated in Figure 3. The first level

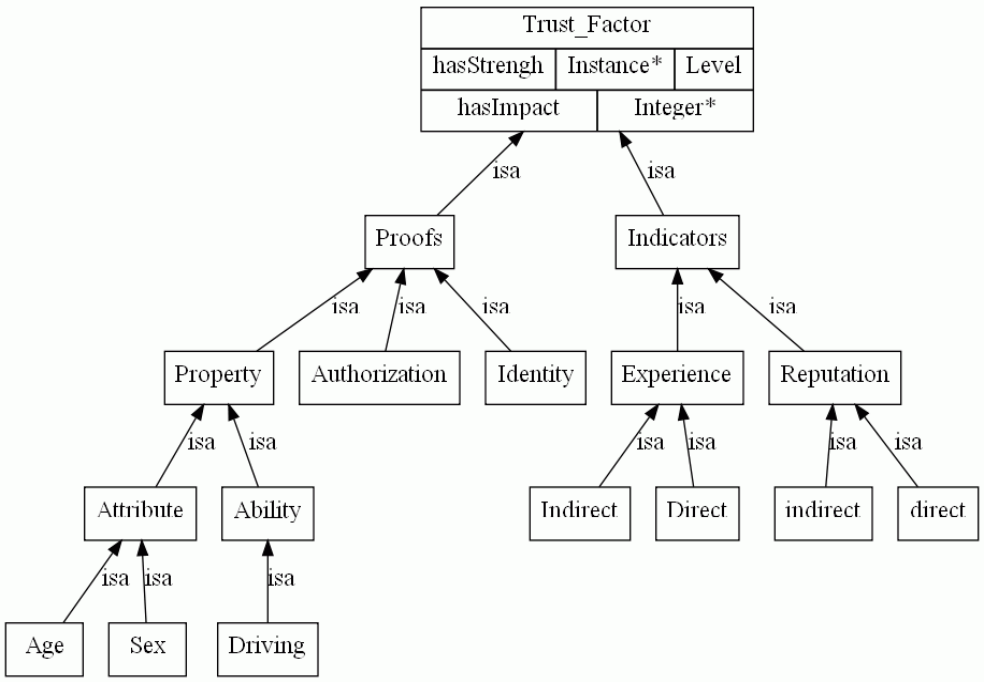

Figure 3: The Trust Factors Ontology

root concept represents a generic trust factor type, it is further divided in two sub-types *proofs* and *indicators*: proofs represent all digitally signed statements, while indicators include the set of all possible facts stored internally or gathered from external sources. Even if they are impossible to check and may be subject to tempering, proofs are valuable hints for trust evaluation in many situations (e.g. when no certification authority exists). For simplicity reasons, we're considering in our ontology, **identities**, **authorizations**, and certified **properties** as proofs, while **experience** (direct and indirect) and **reputation** (direct and indirect) are considered as indicators of trust.

Example 8 (proofs and indicators) *In the running example, agents are using PGP [1] model for the identities management, Authorizations represent permissions controllers' issues and two types of properties are considered, skills and familiarization (e.g. how an agent is skilled in Java and how much he is familiar with Android applications).*

3.3.3 TFO features

As the domains Δ_f of trust factors values are quite heterogeneous (i.e. sets and intervals), and in order to be able to use the same management mechanism for each $f_i \in \mathcal{F}$, we defined a relation $hasStrength : f_i \mapsto l$, where $l \in \mathcal{L}$ is a level representing f_i's strength. While the levels can be as finely grained as desired, we adopte the set of levels $\mathcal{L} = \{VeryLow, Low, Fair, High, VeryHigh\}$. With the same mechanism function $hasImpact : f_i \rightarrow \mathbb{N}^*$ is defined in order to associate to each f_i an impact value representing its weight among the trust decision.

Definition 8 (Trust Criteria and Trust Information) *When $f \in \mathcal{F}$ is used to express a trust requirement we call it a trust criterion (TC). When f is used to state (realease or gather) an information used in order to fulfill a TC, its called a trust Information (TI).*

The ontology $\mathcal{F}$ is used within the model (cf. Fig. 2) in three different ways. (i) For policies specification where $\tau_{\mathcal{F}}$ are used as types and the $l \in \mathcal{L}$ as thresholds for restrictions; (ii) for policies adaptation where $\mathcal{F}$ offers values ordering agents refers to in order to strengthen or weaken their policies; and (iii) while combining individual and social policies where it offers referential values set for choosing the most restrictive policy. Mapping each $f_i \in \mathcal{F}$ enables agents (if needed) not to care about heterogeneity of values and to express trust critetia at abstract level, while being able to map such abstract values with concrete ones at will. These criteria are then checked in comparison with acquired trust information.

Example 9 (Trust Criteria and Trust Information) *Agent a wants to limit the access to the online demo of his service to requesters having a reputation of at least $+0.6$. a states then $TC = (Reputation, +0.6)$ and when he receives an access request r of type $\tau_r = (access, service)$ from an agent b, in order to check whether b satisfies his criterion he needs to acquire a trust information about b's reputation. Hence, when a acquires $TI = (reputation, +0.8)$ about b he considers that his, quite basic, trust requirement is satisfied and may decide to issue the permission $(a, b, \{access\}, \{\})$.*

4 Trust Policies

The trust model represents a specification of trust factors aggregation function. This function is expressed through trust policies, wherein restrictions on types, acceptable values and weights of trust factors are stated (i.e TC expression). This section presents how the previous trust model is concretly expressed in order to implement the semantic of rich, dynamic and combinative trust. It introduces the policy language formalism and describes how policies and Meta-policies are expressed using that formalism.

Definition 9 (Policies) *Let $\mathcal{P} = (\pi \cup \Pi)$ be the set of all policies where π and Π are respectively the sets of all individual and social policies within the system S. $\forall p \in \mathcal{P} = (\pi \cup \Pi), p \in 2^{\tau_{\mathcal{F}} \times \aleph \times \mathbb{N}}$ where $\aleph$ is either a level $l \in \mathcal{L}$ or a value $v \in \mathcal{F}$.*

4.1 Expression formalism

Besides its expressivity and ease-of-use, the required formalism to represent policies must be flexible enough to allow adaptation and combination of policies by adding, removing and updating trust criteria. In order to address such requirements, we adopted first-order predicate structures in order to represents policies. These policies are represented using programming language statements. Prolog is especially interesting as it is declarative and can express a wide range of constraints [6]. For meta-policies, Jason[7], an extension of AgentSpeak [15] (Prolog extension for BDI architectures) has been used. This language offers better practical programming features enabling the combination of declarative and procedural approaches. The most interesting issue in representing meta-policies with this language are plans.

4.2 Representing policies

Policies are represented as a set of TC. Now, let $\mathcal{T} = \{tc_1, tc_1, \ldots, tc_n, \}$ be a set of trust criteria. Each $tc_i, (i \in [1, n])$ is a triplet $\langle \tau_i, \aleph_i, w_i \rangle$. τ_i corresponds to existing trust factors $\tau_f \in \mathcal{F}$, $\aleph_i$ is a threshold value or level from among possibilities, while $w_i \in \mathbb{N}^*$ represents the weight of the tc_i. Let $p_a^t = \{tc_1, tc_2, \ldots, tc_m\}$ be the policy used by the agent $a \in \mathcal{A}$ associated to the request τ_r. The policy is assimilated to a function $f(r, l) = f(p_a^t)$ that evaluates the trust level l relative to an instance request $r \in \mathcal{I}$ of type t. Then :

$$f(p_a^t) = \frac{\sum_{i=1}^{m} f(\tau_i, \aleph_i, w_i)}{\sum_{i=1}^{m} w_i} \tag{1}$$

Where $f(\tau_i, \aleph_i, w_i) \in [0, w_i]$ is the weighted evaluation of the constraint $\aleph_i$ satisfaction on the criterion of type τ_i. Relative importance assigned to each trust criterion is modelled as the weight w_i of a criterion within the policy. Each criterion is then evaluated and returns one (1) or zero (0), respectively, whether the criterion is fulfilled or not. This result is then multiplied by the w_x which is the weight associated to tc_i and returned by f. This policies representation is used to represent both individual and social policies. $\forall p_i^t \in \mathcal{P}$, if $i \in \mathcal{A}$ then p_i^t is an individual policy and belongs to π, if $i \in \mathcal{C}$ then p_i^t is social policy and belongs to Π.

Example 10 (individual and social policies) *Let $r = (alter, object)$ be a request type. The individual policy agent a uses in order to issues permissions for requests of type r is stated as follow: $p_a^r = \{\langle identity, trusted, 2\rangle, \langle$ reputation $, High, 1\rangle\}$. While community c to which a belongs obliges its*

members to use the following policy in handling requests of type $r : p_a^r = \{\langle identity, VeryHigh,\ 3\rangle, \langle reputation, +0.5, 1\rangle\}$.

4.3 Representing Meta-policies

As mentioned in previous sections, meta-policies state how a policy could be changed to handle a specific context. The meta-policies presented in this report are specified by means of AgentSpeak plans. Plans could be described by means of an informal pseudo-code under which each plan is a sequence of event-condition-action rules of the form: $Upon\langle event\rangle : If\langle condition\rangle \leftarrow Do\langle action\rangle$ where $\langle event\rangle$ represents a triggering events, $\langle condition\rangle$ is a general expression, typically a conjunction or a disjunction of literals to be checked before the execution of the plan. Conditions filters defined over the social context, the agent preferences, the requester or the requester resource, while the $\langle action\rangle$ is one or more adaptation actions specified by the agent in order to adapt its policy. The key feature in using Jason plans lies in the possibility to execute legacy code through internal actions. In this work, we use internal action to execute adaptation operations presented in our trust model (for more details on AgentSpeak plans see [15]). The result of the execution of each meta-policy affects the original policy by adding, removing, changing restricting or relaxing one or more trust criteria. The adaptation process will be described with more details in the next section.

Example 11 (Meta-policies) *The agent a is particularly vigilant in regard to the number of agents using his service and the risk associated to such use. He wants then to express adaptation meta-policies in order to adapt dynamically his policy depending on this information. Let r be an access request on the service andro, where* $\tau_t = (access, service)$ *and* p_a^r *be the policy a uses to handle such request from an agent b. Meta-policies he may state are as follow:*
$Adapt(p_{a_1}^r, a_2, andro) : Number(users, X)\&Threshold(users, Y) \wedge (X < Y) \leftarrow .relax(reputation).$
$Adapt(p_{a_1}^r, a_2, andro) : Risk(randro, X)\&Threshold(risk, X)\wedge(X > Y) \leftarrow .restrict(reputation), .add(\langle$"$identity$"$,$ "$Trusted$"$, 3\rangle.)$
The first meta-policy lowers the required reputation value depending on the users amount while the second increases it and adds an identity creteria when the risk is over a certain threshold. Both meta-policies has been express using Jason plans.

5 ASC-TMS : An Adaptive and Socially Compliant TMS

This section presents essential components of our Socially Compliant Trust Management System (ASC-TMS) that assists agents (each agent is equipped

with a personal ASC-TMS) in their trust decision-making process. ASC-TMS is an on-going project that implements the adaptive and combined trust model presented in this report.

5.1 Framework design

The abstract architecture depicted in Figure 4, illustrates our proof-of-concept implementation. We use the JACAMO platform [16] to leverage the requirements discussed earlier while preserving the generality of the approach. The architecture is essentially composed by five main modules and each modules is responsible of a specific operation.

As the principal objective of the framework is the management of trust

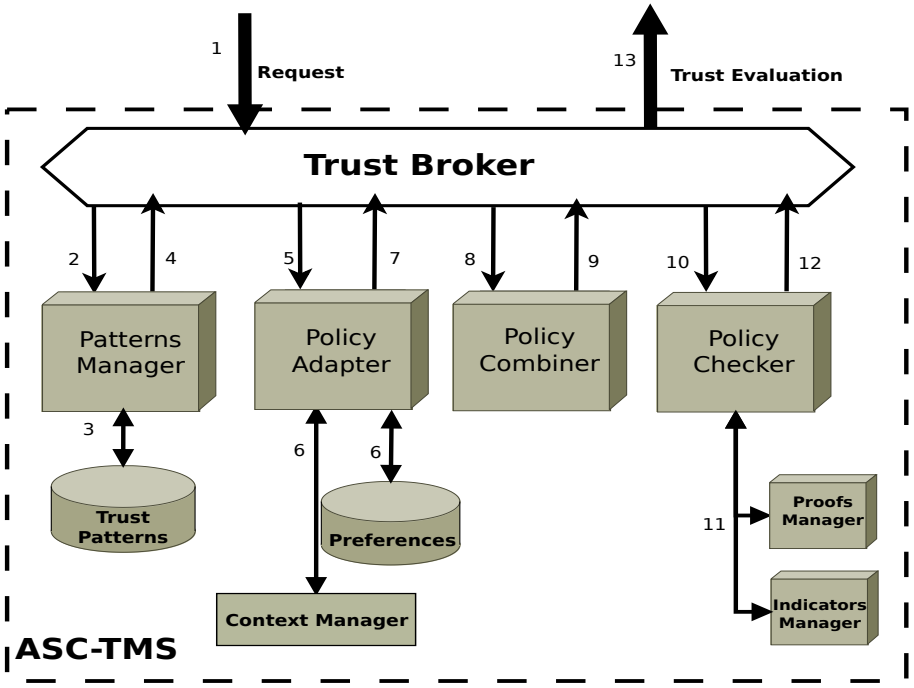

Figure 4: ASC-TMS Framework Architecture

policies by realising the operations presented within the model, we decided to present the framework in respect to its functional features rather then describing the components. The focus will be on essential operations while the others will be briefly described. **Selection** (1-4) operations are realised by the Trust Broker module where the appropriate trust pattern to each request is retrieved from the Patterns Manager based on a mapping matrix ($\tau_q \rightarrow \tau_{\mathcal{O}} \times \tau_{\mathcal{R}}$) **Management** (3) realises the creation and the maintenance of TPs. It enables the user to specify for each request type, the individual policy and associated meta-policies. Social policies are automatically

retrieved from the community and integrated to the TP. The maintenance is processed when one of the policies needs to be updated.

Function Combine($p_a^{\mu,\rho}$,$p_c^{\mu,\rho}$)

$SP \leftarrow p_c^{\mu,\rho}$; $IP \leftarrow p_a^{\mu,\rho}$; $TP \leftarrow \emptyset$; $i \leftarrow 0$
while $SP \neq \emptyset$ **do**
  foreach $SP.tc_x \in SP$ **do**
    if $tc_x.\tau_x \notin IP$ **then**
      $TP.tc_i \leftarrow tc_x$
      $SP \leftarrow SP - \{tc_x\}$
      $i \leftarrow i + 1$
while $IP \neq \emptyset$ **do**
  foreach $IP.tc_x \in IP$ **do**
    if $tc_x.\tau_x \notin SP$ **then**
      $TP.tc_i \leftarrow tc_x$
      $IP \leftarrow SP - \{tc_x\}$
      $i \leftarrow i + 1$
forall the $tc_x \in SP$, $tc_y \in IP$ *s.t.* $tc_x.\tau_x = tc_y.\tau_y$ **do**
  $TP.tc_i \leftarrow \langle \tau_i, max(tc_x.\nu_x, tc_x.\nu_y), max(tc_x.w_x, tc_y.w_y)) \rangle$
  $SP \leftarrow SP - \{tc_x\}$
  $IP \leftarrow IP - \{tc_y\}$
  $i \leftarrow i + 1$
return SP

Adaptation (5-7) is carried out by the Policy Adapter which integrates five main operations : `Add, Remove, Set, Relax, Restrict`. Each operation implements an internal action that is triggered by a meta-policy. The Policy Adapter regularly updates its beliefs-base. When the condition of a meta-policy holds the plan is executed and specified actions are performed. Adaptation is then achieved by the execution of one or more of the following internal actions: $.add(\langle t_i, \nu_i, w_1 \rangle)$: this action adds the criteria $\langle t_i, \nu_i, w_1 tc_i \rangle)$ in the current policy. If another TC of the same type exists, a $.set(\langle t_i, \nu_i, w_1 tc_i \rangle)$ is triggered otherwise. $.remove(t_i)$: the actions removes all TC of the type t_i. t_i is either a specific type as the identity or a general one such as proofs or indicators. $.set(\langle t_i, \nu_i, w_1 \rangle)$: this action affects t_i and ν_i, w_1, respectively, to the strength and the weight of the TC of type t_i in the current policy. If such TC does not exists, an $.add(\langle t_i, \nu_i, w_1 \rangle)$ is triggered. $.restric(t_i)$ / $.relax(t_i)$: These two actions are quite similar. When no parameter is given, all TC of the policy are affected by the action. Otherwise only the specified trust criterion is affected. This action checks the TFO for a lower (resp. higher) value for the specified TC. If such value exist, a set with the new value is called other wise the weight of the TC is lowered

(resp. incresed). If the weight becomes null, the TC is removed.

Combination (8-9) consistes in combining the individual policy along with the social one.

Definition 10 (Combining Policies) *Let $\mathcal{P}_a$ be the policies set of the agent a, $\forall p_a \in \mathcal{P}_a$, p_a is dedicated to a specific request type $\tau_q \subset (\tau_{\mathcal{O}} \times \tau_{\mathcal{R}})$. Let $t \in \tau_q$, p_a^t is the policy the agent a generates to handle a request of the type $t = (\mu, \rho)$ within the community c and $p_a^{\mu,\rho} \in \pi_a^{\mu,\rho}$, $p_c^{\mu,\rho} \in \Pi_c^{\mu,\rho}$ are, respectively individual and social policies set dedicated to the request type t. $p_a^t = p_a^{\mu,\rho} \oplus p_c^{\mu,\rho}$ means that the policy p_a^t is obtained by combining $p_a^{\mu,\rho}, p_c^{\mu,\rho}$ while p_a^t the resulting policy is at least as restrictive as $p_a^{\mu,\rho} \cup p_c^{\mu,\rho}$*

When the policy combiner receives the policies to be combined. It executes the heuristic sketched above (cf. Combine).

Evaluation (10-12) is realised by the Policy Checking module. This part of the framework behaves as basic Trust Management System in a way that it evaluates the policy satisfaction by confronting policy TCs to received and/or gathered TI. When this agent receives a policy to check (10), and after parsing it, it identifies the types of the TC. Proofs are requested from the Proofs Manager (11), while indicators are collected by the indicators manager (11'). When all TIs has been received, the module executes the function described in the previous section (cf. section 4.2) in order to compute all verified TC. The final result is a quantitative evaluation stating the amount of satisfied criteria out of the initially formulated ones.

6 Conclusion And Future Works

We introduced in this report the notion of social compliance for trust management within virtual communities. In order to realize it, we developed a rich, dynamic and combinative model that mimics trust features (i.e. subjectivity, dynamics and context awareness) and preserves real world semantics of trust. We began by exploring and isolating commonly used trust factors from the literature in order to enhance subjectivity. We outlined then a new trust model where individual requirements are adapted and merged with social ones. The originality of this proposition lies in the use of dynamic and adaptive trust policies (and meta-policies) in order to capture and represent the dynamisms of trust. Finally, we made the first steps towards designing a Socially Compliant Trust Management System (ASC-TMS) based on Multi-agent technologies.

The next steps in our work include extending the model in order to integrate individual trust requirements (policies) into social ones. The idea would be to study how agents may influence / trigger adaptation or evolution of social policies. We also plan to investigate mechanisms to automatically build social policies from individual ones. Such feature is particularly interesting for

decentralized and self-organized communities like social networks (diaspora for example).

References

[1] A. Abdul-rahman. The PGP Trust Model. *Architecture*, pages 1–6, 1997.

[2] P. Anantharam, C.A. Henson, K. Thirunarayan, and A.P. Sheth. Trust Model for Semantic Sensor and Social Networks : A Preliminary Report. *Scenario*.

[3] D. Artz and Y. Gil. A survey of trust in computer science and the semantic web. *Web Semant.*, 5:58–71, June 2007.

[4] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. *In Proceedings of IEEE Symposium on Security and Privacy*, 1996.

[5] P. Bonatti, C. Duma, D. Olmedilla, and N. Shahmehri. An Integration of Reputation-based and Policy-based Trust Management. *Management*.

[6] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *Proceedings of the 7th ACM conference on Computer and communications security*, CCS '00, pages 134–143, New York, NY, USA, 2000. ACM.

[7] R.H. Bordini and J. Hübner. Semantics for the jason variant of agentspeak. In *Proceeding of the 2010 conference on ECAI 2010*, pages 635–640, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[8] R. Falcone and C. Castelfranchi. *Social trust: a cognitive approach*, pages 55–90. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[9] T. Gruber. Toward Principales for Design of Ontologies Used for Knowledge Sharing, 1993.

[10] A. Herzberg, Y. Mass, J. Michaeli, Y. Ravid, and D. Naor. Access control meets public key infrastructure In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, 2000.

[11] A. Jøsang. Foundations of security analysis and design iv. chapter Trust and reputation systems, pages 209–245. Springer-Verlag, Berlin, Heidelberg, 2007.

[12] Karl K., Mogens N., and Vladimiro S. A framework for concrete reputation-systems with applications to history-based access control. In *In Proc. of the 12th CCS*, pages 7–11, 2005.

[13] A. J. Lee, T. Yu, and Y. Le Gall. Effective trust management through a hybrid logical and relational approach. pages 169–179, 2010.

[14] D.W. Manchala. E-commerce trust metrics and models. *IEEE Internet Computing*, 4:36–44, March 2000.

[15] A. Rao. AgentSpeak (L): BDI Agents speak out in a logical computable language. (L), 1996.

[16] A. Ricci, J.F. Hubner, R. H Bordini, and O. Boissier. JaCaMo Project, 2010.

[17] J. Sabater and C. Sierra. Regret: A reputation model for gregarious societies. pages 61–69, 2001.

[18] K. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu. Requirements for policy languages for trust negotiation. In *Proceedings of POLICY '02, USA, 2002.*

[19] Li X. and Ling L. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 16:843–857, 2004.

[20] J. L. De Coi, D. Olmedilla A Review of Trust Management, Security and Privacy Policy Languages *International Conference on Security and Cryptography (SECRYPT 2008). INSTICC Press*, July 2008.